AMENDMENTS TO THE SPECIFICATION:

Please replace the paragraph [0023] on page 6 through page 7, with the following amended paragraph:

[0023] In the present illustrated examples, the process of establishing a connection in accordance with HTTP will be considered. Usually a request for such a connection is made by the web browser application program, and this in turn is most likely to be at the behest of a user operating the web browser. Where this is the case, the request will identify the address or "URL" within the network of the computing entity with which a connection is sought, initially using alphanumeric characters entered at the address bar of the browser application program ~~(for example http://www.hp.com)~~. Ultimately however these are "resolved" into a numerical "IP address" of the form: xxx.xxx.xxx.xxx, where xxx is an integer between 0 and 255 inclusive. An example of an IP address is 192.168.2.2. The IP address is subsequently further resolved into what is known as a physical, or Media Access Control ("MAC") address of the network card of the destination computing entity. Resolution of the URL into an IP address, and the IP address to a MAC address usually takes place at dedicated computing entities within the network, in a manner which is well known per se, and will not be described further herein. This description of the connection process in accordance with HTTP, well known per se, has described connections legitimately requested by a user, and by means of a URL. However it should be appreciated that it is possible for example to request a connection from the web browser application program using an IP address, rather than the alphanumeric characters of the URL. This is an aspect of the system behaviour which has been exploited by viruses, some of which randomly generate IP addresses in accordance with the rules governing their allowable format, and then seek connection to those randomly generated addresses.

Please replace the paragraph [0033] on page 13, with the following amended paragraph:

[0033] Because receipt of requests are the trigger for the commencement of the routine illustrated in Fig. 7, neither the number of occasions in a given time window in which the VPMS routine is run, nor the timing of their commencement can be known in advance. Additionally, as illustrated in Fig. 7, it is possible for two (or indeed more, although only two are

illustrated in Fig. 7) routines to be running in temporal overlap, since one may still be running when another is triggered by a further request. Similarly, a request may trigger the execution of the routine of Fig. 7 just prior to the end of a time window (a situation also illustrated in Fig. 7, with steps which occur at the end 720 of a time window/the beginning 702 of a subsequent time window being shown in dashed lines), so that the execution of the routine may overlap temporally with a part of the next time window. The approach taken by this particular embodiment to this issue of overlap is relatively simple: if at the commencement of time window T_{n+1}, the update of the dispatch record for a previous time window T_n has been completed during the simultaneous running of a VPMS routine commenced in the previous time window T_n, but prior to execution the step 712 (adding a request to the virtual buffer) for that routine, the subsequent update of the virtual buffer in that step 712 will be treated as if performed for a request received in the current time window T_{n+1}. This approach has the benefit of being simple, although it may on occasions yield minor inaccuracies, with a request being recorded as being outside of the policy simply because processing of the request received and initially processed during one time window extended into the next time window, but this is not significant overall.

Please replace the paragraph [0062] on page 30 through page 32, with the following amended paragraph:

[0062] Referring now to Fig. 13A, a plot of activity (i.e. the number of requests processed by the VAPS) against time is illustrated for example of Fig. 10A. From this graph it can be readily appreciated that prior to the viral infection signified by the rapid increase in the number of requests during the time interval T4, only a relatively low number of requests are processed per time interval, and that therefore it is possible to use the VAPS to carry out a policy preventing connection to more than one new host per time interval without impeding legitimate network traffic to any significant extent. Consider however an excerpt of a graph illustrating legitimate traffic flow in Fig. 13B, where there are significant levels of activity, interspersed by a much shorter period of time during which there is no activity at all. Applying the rather simple policy of permitting connection to one new host per time interval, where all time intervals are of the same duration would significantly impede the flow of the legitimate network traffic

illustrated in Fig. 13B. Ideally therefore, an alternative policy is required which accounts for the nature of this legitimate traffic flow. An example of such a policy is illustrated referring now to Fig. 13C, where two sorts of time intervals are illustrated: S_l, a relatively long time interval, and S_s, a relatively short time interval. From Fig. 13C it can be seen that when placed together alternately, the time intervals S_l corresponds to the time interval in the graph of the traffic flow from Fig. 13B where there is a flow of traffic, and the time interval S_s to the time interval between two such time intervals, where there is no traffic flow. By segmenting time for a VAPS using these two time intervals therefore, it is possible to construct a policy which matches closely the legitimate behaviour illustrated in Fig. 13B, but still provides an impediment to the propagation of viruses. Such a policy for the VAPS may be implemented using the variable LogNo, which as explained above corresponds to the number of requests present in the delay buffer at the end of any given time interval. In the present example it is desirable to implement a policy which does not impede the free flow of the legitimate traffic pattern illustrated in Fig. 13C, and referring now to Fig. 14, to this end a modified first buffer management routine is provided. Following a clock timeout at step 1402, the routine determines at step 1404 whether the LogNo is greater than a predetermined number, in this instance 10, this number being chosen, in conjunction with the number of request identities held in the dispatch record, to be equal or slightly larger than the number of requests typically received during a "long" time interval S_l. If LogNo is greater than this number, then the routine defaults to step 1408, where it transmits only the first request in the delay buffer, and then proceeds to steps 1412 to 1416 where identical requests are transmitted (step 1412) the record is updated (step 1414), and the value of LogNo is updated (step 1416). If LogNo is less than 10, i.e. less than 10 new requests have been received during the course of that time interval, then the routine proceeds to step 1406, at which it determines whether a further variable LogNoLast, equal to the number of new requests received during the previous time interval, is greater than zero. If it is, then the routine defaults once again to step 1408 where only a single request is transmitted from the delay buffer. If it is not, i.e. no new requests were received during the previous time interval, then the routine acts to transmit, at step 1410, requests 1-10 from the delay buffer, followed by the steps 1412 to 1416. Thus, when 10 or less new requests are received during a time interval, and no new requests were received during the previous time window, the routine operates to transmit all 10 requests.

This mimics the legitimate activity during a "long" time interval S_l, where the activity level is relatively high, but in the previous short time interval activity was zero. Correspondingly, in any time window where there were more than 10 new requests (i.e. a greater level of activity than usual in a longer time interval) or where, in the previous time window there were more than zero new requests (which is the pattern of legitimate traffic flow illustrated in Fig. 13B), the routine defaults to what may be thought of as the "standard" policy of one new request per time interval, thus throttling activity differing from usual legitimate activity, and which is likely to be virally-related. The modified routine thus carries out a policy which conforms generally to the behaviour pattern illustrated in Fig. 13C.